**Day 7**

**What to do?**

Learn about gradient descent and its variants.

The goal of ANY ML algorithm is to have a model with low cost. In other words, the gradients to reach their local optima which produces low cost. For some algorithms, it might be a straight descent path, for some it might be noisy. Whether it is straight or noisy, gradient descents and learning rate makes sure that the loss does not get stuck midway.

When it comes to DL/NN, backpropagation and gradient descent work together to give us the best model for the problem statement. Given the error that was calculated during forward prop for the network, the networks perform error calculation backwards too. So, the gradient of the network depends on the gradient of pervious layers. There are three formats (as far as I know of) to perform gradient descent (for any ML algorithm, actually!).

1. **Batch gradient descent:**

This technique takes in entire batch of training dataset at once to compute its gradients. Even though this maintains accuracy, the model will be terribly slow (especially if the size of the data is super large).

In Keras, this can be done by setting *batch\_size* to *len(training\_samples)*.

1. **Mini-batch gradient descent:**

Mini-batch gradient descent is remarkably like batch gradient descent. The only difference is that in mini batch, the training data is split into smaller batches of data. Let us say you have a training dataset of size 10,000 rows. You can have 100 batches each containing 100 rows.

Basic rule of mini batch, you set the batch size to be greater than 1 and less than length of training examples. According to Ruder, acceptable batch size is between 50 and 256 samples. Compared to batch gradient descent, mini batch is faster and less variant.

1. **Stochastic gradient descent:**

Last approach is to compute VERY SMALL steps, i.e., batch size is set to 1 sample. This is extremely fast method, but performance becomes at stake.

